Descripción de los Lenguajes Aceptados por Autómatas

Los Teoremas de Kleene

Universidad de Cantabria

Esquema

Lenguajes Aceptados por Autómatas

Como repaso, tenemos un problema de respuesta “Si/No” que hemos conseguido resolver utilizando un autómata finito.

- ¿Como es el conjunto de soluciones del problema?
- ¿Qué problemas se pueden resolver con autómatas?

Lenguajes Aceptados por Autómatas

Como repaso, tenemos un problema de respuesta “Si/No” que hemos conseguido resolver utilizando un autómata finito.

- ¿Como es el conjunto de soluciones del problema?
- ¿Qué problemas se pueden resolver con autómatas?

Lenguajes Aceptados por Autómatas

Demostraremos que los lenguajes aceptados por autómatas finitos son los lenguajes regulares. Y el recíproco también es cierto.

Teorema de Análisis de Kleene

Teorema

Sea $L \subseteq \Sigma^*$ un lenguaje aceptado por un autómata finito determinista. Entonces, existe una expresión regular α sobre el alfabeto Σ tal que $L = L(\alpha)$. Más aún, mostraremos que existe un algoritmo que permite calcular la expresión regular asociada al lenguaje aceptado por un autómata.

Algoritmo

Para demostrar este teorema construiremos un sistema de ecuaciones lineales en expresiones regulares con las reglas siguientes:

Algoritmo

Supongamos que $Q := \{q_0, \ldots, q_n\}$.
Introducimos un conjunto de variables biyectable con Q dado por $\{X_0, \ldots, X_n\}$. La biyección será dada por $q_i \longmapsto X_i$.

Algoritmo

Definimos un sistema de ecuaciones lineales en expresiones regulares:

$$\begin{pmatrix} X_0 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} \alpha_{0,0} & \cdots & \alpha_{0,n} \\ \vdots & \ddots & \vdots \\ \alpha_{n,0} & \cdots & \alpha_{n,n} \end{pmatrix} \begin{pmatrix} X_0 \\ \vdots \\ X_n \end{pmatrix} + \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_n \end{pmatrix},$$

Algoritmo

- Para cada i, $0 \leq i \leq n$, definamos $\beta_i = \lambda$ si $q_i \in F$ y $\beta_i = \emptyset$ si $q_i \notin F$.
- Para cada i, j, $0 \leq i, j \leq n$, definamos $A_{i,j}$ mediante:

$$A_{i,j} := \{z \in \Sigma \ : \ \delta(q_i, z) = q_j\}.$$

Definiremos

$$\alpha_{i,j} := \sum_{z \in A_{i,j}} z,$$

notando que si $A_{i,j} = \emptyset$, entonces, $\alpha_{i,j} = \emptyset$.

Algoritmo

Si $(\alpha_0, \ldots, \alpha_n)$ es una solución del anterior sistema lineal, $L(\alpha_0)$ es el lenguaje aceptado por el autómata.

Teorema de Síntesis

Teorema

Dado un lenguaje L aceptado por un autómata, existe un autómata $A := (Q, \Sigma, q_0, F, \delta)$ *que acepta L y que verifica las siguientes propiedades:*

1. $\sharp(F) = 1$, *es decir, sólo hay una configuración final aceptadora. Supondremos* $F := \{f\}$.
2. $\delta(q, x)$ *está definida para todo* $q \in Q$ *y todo* $x \in \Sigma$.
3. *Las únicas* $\lambda-$*transiciones entran en f. Es decir,*

$$\text{Si} \;\; \delta(p, \lambda) = q \Leftrightarrow q = f.$$

Idea de la construcción

Dado el autómata $A := (Q, \Sigma, q_0, F, \delta)$, que podemos suponer determinista, veamos ahora como construir el nuevo autómata $\bar{A} := (\bar{Q}, \Sigma, q_0, \bar{F}, \bar{\delta})$.

Idea de la construcción

Sea $f, ERROR$ dos nuevos estados tal que $f, ERROR \notin Q$. Definamos $\bar{Q} := Q \cup \{f\} \cup \{ERROR\}$. El único estado final sera f.

Idea de la construcción

El estado *ERROR* será un estado de fallo y añadimos $\lambda-$transiciones de los antiguos estados finales al nuevo estado final *f*. Todas las transiciones no definidas van al estado *ERROR*. Es claro que el nuevo autómata acepta el mismo lenguaje que el antiguo.

Teorema de Síntesis de Kleene

Teorema

Sea Σ un alfabeto finito y α una expresión regular sobre Σ. Entonces, existe un autómata finito A que reconoce el lenguaje $L(\alpha)$ descrito por α. Más aún, el proceso de obtención del autómata a partir de la expresión regular se puede lograr de manera algorítmica.

Algoritmo

El caso de los símbolos primarios:

- *El caso* $\emptyset$*:* Bastará un autómata con $Q := \{q_0, q_1\}$, $F := \{q_1\}$ tal que la función de transición no esté definida en ningún caso.
- *El caso* λ*:* De nuevo usaremos $Q := \{q_0, q_1\}$, $F := \{q_1\}$, pero la función de transición está definida solamente para $\delta(q_0, \lambda) = q_1$ y no definida en el resto de los casos.
- *El caso constante* $a \in \Sigma$*:* Igual que en el caso anterior, usaremos $Q := \{q_0, q_1\}$, $F := \{q_1\}$, pero la función de transición está definida solamente para $\delta(q_0, a) = q_1$ y no definida en el resto de los casos.

Algoritmo

El autómata de la unión ($\alpha + \beta$): Si tenemos $A_1 := (Q_1, \Sigma, q_1, F_1, \delta_1)$ un autómata determinista que acepta $L(\alpha) \subseteq \Sigma^*$ y un segundo autómata también deterministas $A_2 := (Q_2, \Sigma, q_2, F_2, \delta_2)$ un autómata que acepta $L(\beta) \subseteq \Sigma^*$, definimos un nuevo autómata que refleja la actuación de los dos autómatas al mismo tiempo.

Algoritmo

Viene dado por las reglas siguientes:

- $Q := Q_1 \times Q_2$,
- $F := (F_1 \times Q_2) \cup (Q_1 \times F_2)$
- $Q_0 := (q_1, q_2)$
- $\delta((p, q), z) = (\delta_1(p, z), \delta_2(q, z))$, $\forall p \in Q_1, q \in Q_2$ y $\forall z \in \Sigma \cup \{\lambda\}$.

Algoritmo

El autómata de la concatenación ($\alpha \cdot \beta$)*:* Supongamos $A_1 := (Q_1, \Sigma, q_1, F_1, \delta_1)$ un autómata que acepta $L(\alpha) \subseteq \Sigma^*$ y un segundo autómata $A_2 := (Q_2, \Sigma, q_2, F_2, \delta_2)$ un autómata que acepta $L(\beta) \subseteq \Sigma^*$. En necesario que A_1 verifica que solo tiene un estado final, además que esta definido para todos los estados y solo hay pocas $\lambda-$transiciones.

Algoritmo

- $Q := (Q_1 \times \{1\}) \cup (Q_2 \times \{2\})$.
- $F := F_2 \times \{2\}$.
- $q_0 := (q_1, 1)$
- La función de transición $\delta : Q \times (\Sigma \cup \{\lambda\}) \longrightarrow Q$, viene dada por:

$$\delta((q,i),z) := \begin{cases} (\delta_1(q,z),1), & \text{si } q \in Q_1,\ i = 1 \\ (q_2,2), & \text{si } q = f \in F_1,\ i = 1,\ z = \lambda \\ (\delta_2(q,z),2), & \text{si } q \in Q_2,\ i = 2 \end{cases}$$

Algoritmo

El autómata del monoide generado (α^*): De nuevo suponemos que tenemos un autómata $A := (Q, \Sigma, q_0, F, \delta)$ que acepta el lenguaje $L(\alpha)$.

Algoritmo

- Para cada $q \in Q \setminus F$ y para cada $z \in \Sigma \cup \{\lambda\}$, definamos $\bar{\delta}(q, z) := \delta(q, z)$.
- Finalmente, definamos:

$$\bar{\delta}(f, \lambda) := q_0.$$

y

$$\bar{\delta}(q_0, \lambda) := f.$$

Método Adicional

Inicialmente, empezamos dibujamos dos estados, el primero sera el inicial y el segundo el final. Dibujamos una transición con la expresión regular de la cual queremos hallar el autómata.

Método Adicional

Repetir el los siguientes pasos hasta que todas las transiciones no contengan más que símbolos del alfabeto.

- Remplazar cada transición donde aparezca $f + g$ por dos transiciones, entre los mismos estados con f y g.
- Añadir un nuevo estado por cada transición donde aparezca fg y añadir dos transiciones, la primera que vaya del estado inicio hasta el nuevo nodo con f y la segunda que vaya del nuevo nodo al estado antiguo por g.
- Añadir también un nuevo estado por cada transición donde aparezca $f*$, añadiendo $\lambda-$transiciones que entren y salgan al nuevo estado de los estados antiguos y una transición del nuevo estado en si mismo con "símbolo" f.